In The Specification:

Please implement the following changes in the specification. Specifically, please implement the following change in paragraph **[0051]**:

[0051] Nodes in a B-Tree may contain handles referring to other nodes. In most B-Tree variants, the handles connect the nodes to form a tree (hence the name), a directed, connected, and acyclic graph. It is noted that every tree is a directed acyclic graph, but not every directed acyclic graph is a tree. For instance, a B-Link Tree, discussed below, is a directed acyclic graph, but not a tree. In the following, the reader is assumed to be familiar with the definition of a tree and the terms subtree, link, root, leaf, parent, child, and sibling. B-Link Trees differ from proper trees in that in addition to the links from parents to children, every node has a link to its directly adjacent right sibling (if such a sibling exists). This can be seen in the exemplary B-Link Tree ~~300~~ 206 of Fig. 2B, where the "right link" (link to right sibling) is represented by reference numeral ~~302~~ 216.

Please implement the following changes in paragraph **[0052]**:

[0001] A B-link tree can be thought of as composing two different kinds of nodes: data nodes and index nodes, reference numerals ~~304~~ 214 and ~~306~~ 212 , respectively, of Fig. 2B. A data node is simply a key-value pair of the form $\langle k,d \rangle$. An index node is of the form: :

$\langle k_{\min}, h_0, k_0, h_1, k_1, \ldots, h_{n-1}, k_{n-1}, h_n, k_{\max}, h_{\text{right}} \rangle$

Please implement the following changes in paragraph **[0061]**:

[0002] With respect to an infinite persistent log, logging for a single server is first considered. In this scenario, it is ideally assumed that an infinite persistent log is available, i.e., that a log exists that is persistent and has no bounds/memory constraints. Fig. 3A depicts an information retrieval system 300 that utilizes an infinite persistent log 312. As shown, such a system can include a server 302 and a persistent store 304, such as a database. In addition, the data residing in the store 304 may be organized in the form of a tree, e.g., a B-link tree 306. Such a data structure includes nodes, N1, N2, N3 and so on, and, in the case of index nodes, links from each node to at least one other node. While log 312 is depicted as included in persistent store 304, log 312 could be provided in separate persistent storage.

Please implement the following changes in paragraph **[0065]**:

[0065] With respect to caching, Fig. 3B depicts another embodiment of information retrieval system 300. In a typical implementation, server 302 (which may comprise multiple computing devices) includes a B-link tree layer 310 generating log entries 314, a memory 335 and cache memory 330, which raises the question of how to log the operations of the B-link tree layer 310, some of which may be present in cache memory 330 as an intermediate step, i.e., how to handle log entries 314 describing data transactions to be performed on data structure 306 in such a system. Described in more detail below, one way to handle logging is to utilize memory 335 to ~~stores~~ store log entries 314 that are not yet committed to persistent storage 304 while utilizing cache memory 330 to store updated records of data structure 306 as an intermediate step before commission to persistent storage 304.

Please implement the following changes in paragraph **[0066]**:

[0003] Performance is improved in such a system because not all updates from the B-link tree layer 310 need to go directly to persistent storage 304 as changes to data structure 306. Thus, as illustrated in Fig. 3B, if the operations are organized in transactions, the log record updates corresponding to a transaction are written to the persistent storage 304 when the corresponding transaction commits. In the meantime, they can be cached in cache storage 330.